**原型链**

**创建对象的几种方式：**

1.字面量

var o1 = {name: 'o1'};

var o2 = new Object({name:"o2"})

2.通过构造函数

var M = function(name) {

this.name = name;

}

var o3 = new M("o3")

3.第三种方式：Object.create

var p = {name: "o4"}

var o4 = Object.create(p);

延伸（用函数实现new运算符）

var new2 = function(obj) {

var o=Object.create(func.prototype); //用原型创建一个空白对象

var k=func.call(o); // 改变this指向，把this托付给k

if(typeof k === 'Object'){

return k;

}else{

return o;

}

}

ps: new 做了什么？

（1）创建一个新对象；

（2）将构造函数的作用域赋给新对象（因此this就指向了这个新对象）；

（3）执行构造函数中的代码（为这个新对象添加属性）；

（4）返回新对象。

构造函数- prototype- 》原型对象 - constructor - 》构造函数

构造函数 - new - 》实例 - proto - 》 原型对象

**继承**

ES6

创建对象

class Parent {

constructor(name = 'mokewang') {

this.name = name;

}

get longName() { //属性

return 'mk' + this.name;

}

set longName(value) {

this.name = value;

}

}

let p = new Parent("v"); // p{name : 'v'};

class Child extends Parent {

constructor(name = "child") {

super("moke")

this.name = name;

}

}

let c = new Child('s') //c{name:"s"}

extends干了一件什么事？

1、把子类构造函数(Child)的原型（`\_\_proto\_\_`）指向父类构造函数（Parent）

2、把子类实例child的原型对象（child.prototype）的原型(\_\_proto\_\_)执行父类的原型对象(parent.prototype)

3、子类构造函数使用`super`继承父类构造函数属性

**get set**

监听longName的读取和赋值

**静态方法**

在方法前加上static。该方法可正常调用但无法修改。